Funciones en “C”

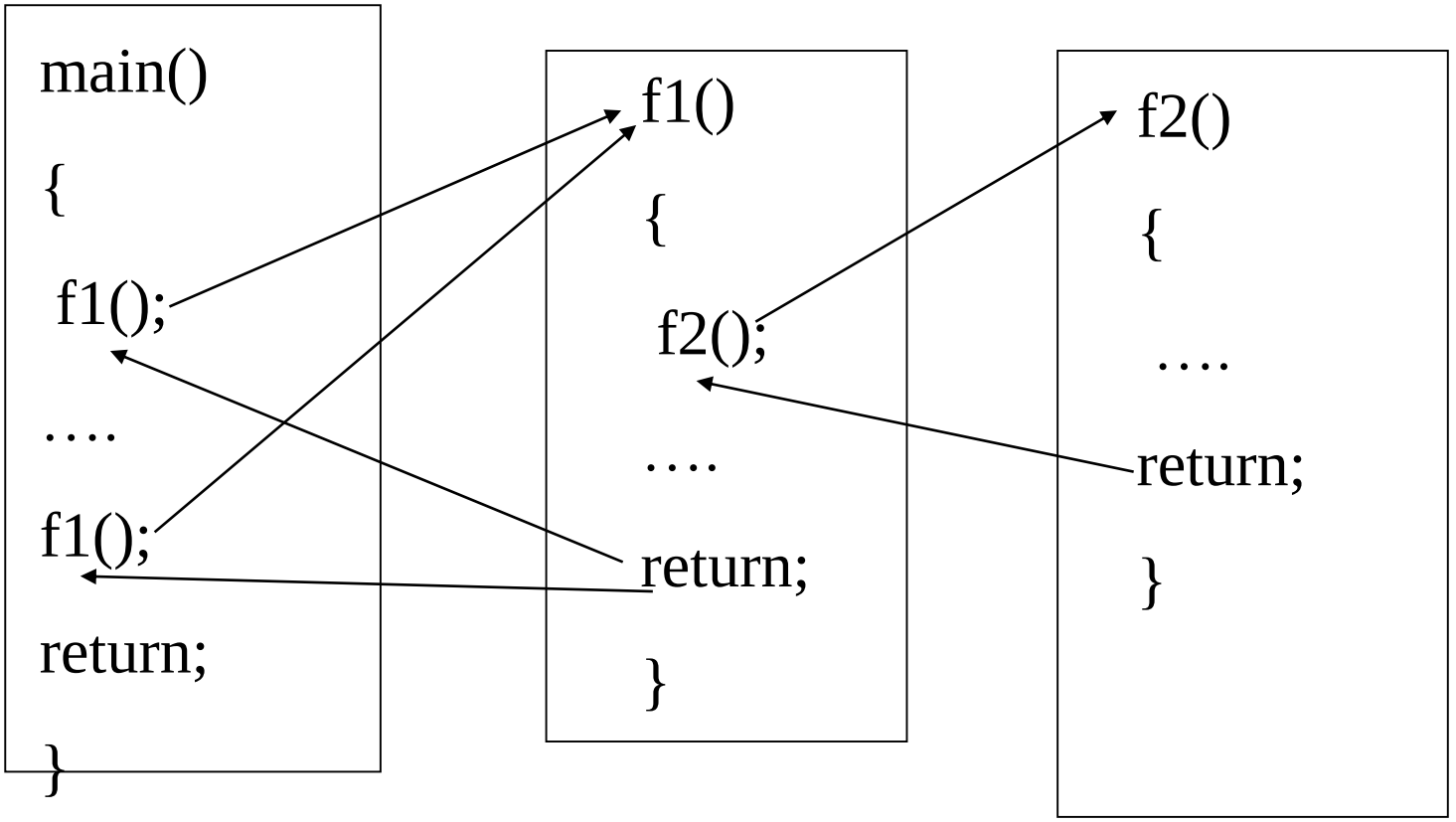
Funciones

- Son bloques con los que se construyen programas en C.

Características

- Permiten dividir un problema en subproblemas mas fáciles de realizar y mantener.
- Todo programa consta de al menos una función

Flujo de control de llamado



Definición

```
tipo nombre([,argumentos])
  {
    [declaraciones]
    proposiciones
    return[(expresión);]
  }
```

Encabezado

- Tipo: El tipo del valor devuelto
- Nombre: Identificador válido
- Argumentos: Declaraciones separadas por ; de argumentos que recibe la función.

- Nota: si no recibe ni devuelve se coloca void
- Por default devuelve un entero

Cuerpo de la función

- Sentencia compuesta que define lo que hace la función.
- Puede contener declaraciones de variables utilizadas en dichas sentencias. Estas variables, por defecto son locales a la función.
- **[return(expresion)]** se utiliza para

Devolver el valor de la función

Ejemplo

```
void letrero(void)
{      /* no variables locales*/
printf("\n Esta es una función muy simple");
return;   /* no devuelve */
}
```

No recibe

No devuelve

Ejemplo

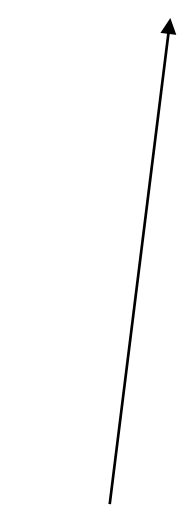
```
void checa_par(int i)
 {    /* No variables locales*/
   if (i>0)
     printf("\n i es positivo");
   else
     printf("\n i es negativo");
   return;   /* no devuelve*/
 }
```

No recibe

No devuelve

```
int suma(int a, int b)
{
   int res;
   res=a+b;
   return(/*a+b*/ res);
}
```

Ejemplo



```
int suma(int a, int b)   Recibe int
   { int valor; /*variable local*/
     valor = a+b;
a=a+20;
     return(valor);
   }
```

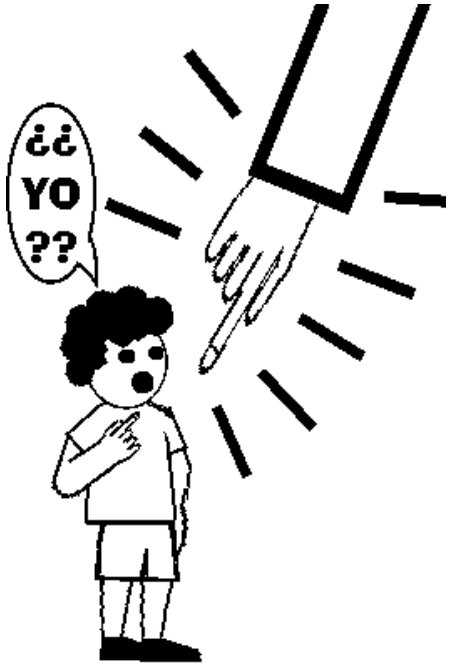
Devuelve int

```
float sueldo(int hrs, float pa_hr)
{ float paga;
  paga=hrs*pa_hr;
  return(paga);
}
```

Llamado

- Se hace mediante su nombre con los argumentos entre paréntesis.

- Generalmente se asigna el valor de la función a una variable del mismo tipo de esta.



Ejemplo

```
#include <stdio.h>
#include <entrada.h>
int main()
 { int res, int x=7, int y=8;
   letrero(); /*no devuelve ni recibe*/
   checa_par(10);   /*no devuelve*/
   res=suma (5,10); /* recibe y devuelve*/
   printf("La suma es : %d\n",res);
   printf("La suma es : %d\n", suma(x,y));
res=suma(10*20+3*80, sqrt(24))
}
```

Declaración de prototipo

- Una función prototipo tiene la misma sintaxis que el encabezado de una función y termina (;).

- Es necesario hacer una declaración forward cuando se hace el llamado de la función antes de haberla definido.

Ejemplo

```
int main()
{
   /* declaración de una función prototito */
  double escribe(double, int);
  double r,a=3.14;
  int b= 5 ;
  r = escribe(a,b); /* llamada a la función */
  printf("%.2lf \n",r);
 }
    /* definición de la función */
 double escribe(double x, int y)
   {
     return( x + y * 13 );
    }
```

Variables Locales

Se declaran dentro de algún bloque.

Solo pueden ser usadas en el bloque donde fueron definidas.

Ejemplo

```
int main (int)
 {
   int x;  /* Es variable local*/
 }
```

Variables globales

- Se declaran al inicio del programa fuera del main() y fuera de cualquier función.

- Pueden modificar su valor en cualquier parte del programa

Ejemplo

```
#include <stdio.h>
int x=20;   /*variable  global*/
void escribe_x()
 {
   printf(“ El valor de x (Global) es = %d\n”,x);
x=x+2;
    return;                              /*x=20 */
  }
main()
 {
   int x=12;                  /*variable local*/
   escribe_x(); /* Escribe el Valor de x = 20 */
   printf(“ El valor de x (Local) es = %d\n”,x);
                                /* x=12 */
}
```

Paso de parámetros

- **Paso de parámetros por valor :**

copiar los parámetros actuales en sus correspondientes *lista de argumentos.*

Sintaxis

var_recibe=nomfun(variable_par)

ejemplo, res=suma(a,b);

- **Paso de parámetros por referencia :**

Lo transferido son las direcciones de las variables que contienen esos valores

Sintaxis

var_recibe=nomfun(&variable_par)

ejemplo, res=suma(a, &b)

Por referencia

Una referencia es un valor que permite acceder de forma indirecta a un dato en particular.

El apuntador es una referencia.

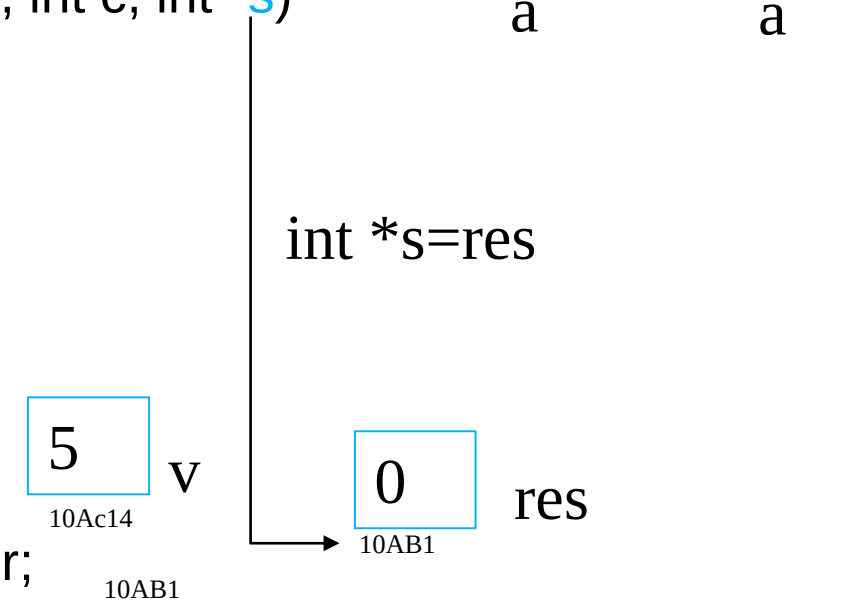
* Operador de desrefencia

```
int num = 5;
int *ptr_num = &num;
printf("valor: %d\n", *ptr_num);  /*Imprime 5*/
```

Ejemplo

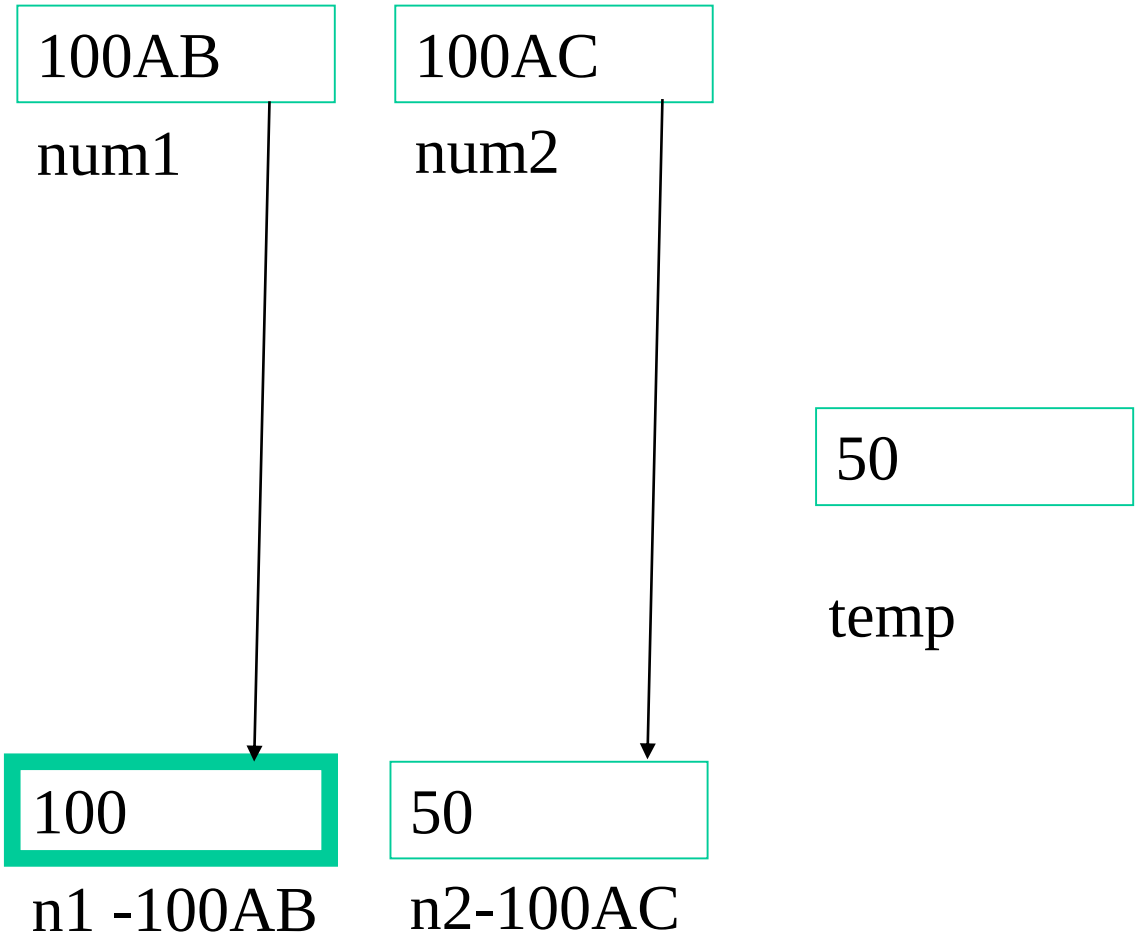
Ejemplo:

```
#include <stdio.h>
/* función sumar*/ 5
int sumar(int a, int b, int c, int *s)
    {int k=5;
      b += 2;
      s = a + b + c;
      return(k);
     }
 int main()
    {
      int v = 5, res=0, r;
       r= sumar(4,v,v*2-1,&res); /* llamado a la función */
      printf("%d",res);   /* 18*/
      return;
    }
```



Ejemplo

```
int compara(int *num1, int *num2, int *menor, float *res)
{
  if(num1>num2)
    { int temp, mayor;
      temp=*num1;
      *num2=*num1;
       *num1=temp;
     mayor=num1;
     *menor=num2;
     *res=mayor+(*menor);
    }
  else
    {
     mayor=num2;
      *menor=num1;
  return(mayor);
```


100AB
num1
100AC
num2
50
temp
100
n1 -100AB
50
n2-100AC

Ejemplo: Pasar una cadena

```
#include<stdio.h>
#include<string.h>

int cad(char *cad)
{
  int i;
  puts(cad);
  for(i=0;i<strlen(cad);i++)
        cad[i]='.';
   return 0;
}

int main(void)
{
 char nif[ ]="hola todos";
 cad(nif);   /*cad(nif[0])*/
 puts(nif);
return 0;
}
```

int a; int b[100];

Func(int *a) Funcion(int *b)

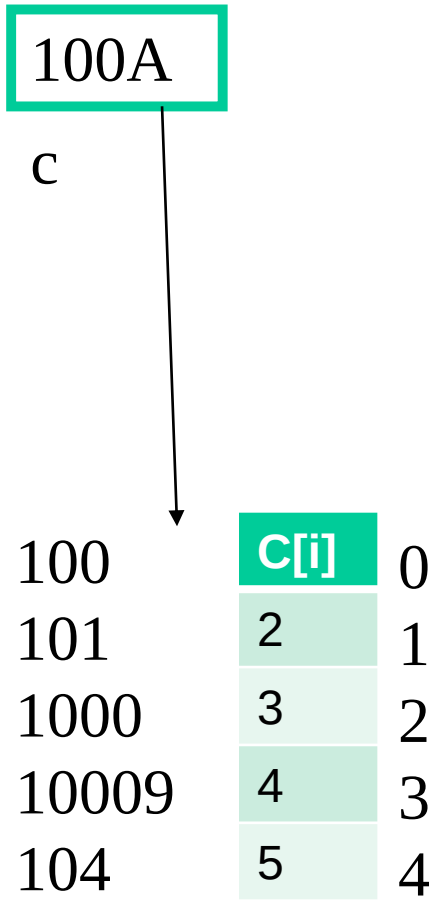
Func(&a) Funcion(struct ife b)

Ejemplo: Pasar vector

```
#include<stdio.h>
#include<string.h>

int mult(int *c)
{
  int i;
  for(i=0;i<5;i++)
        c[i]=c[i]*2;
  return 0;
}

int main()
{
 int i;
 int n[]={1,2,3,4,5};
 mult(n);
 for(i=0;i<5;i++)
   printf("%d",n[i]);
return 0;
}
```



Ejemplo: Pasar una estructura

```
#include<stdio.h>
#include<string.h>

struct ficha
 {
   char *nombre;
   char *apellido;
   int edad;
   float salario;
}mi_ficha[2];
```

```
void muestra(struct ficha *p);
int main()
{
  struct ficha *st;
  int c;
st=mi_ficha;
mi_ficha[1].apellido=“lop”;
mi_ficha[1].nombre=“rob”;
mi_ficha[1].edad=30;
mi_ficha[1].salario=345.69;
muestra(st /* mi_ficha*/);
return 0;
}
```

```
void muestra(struct ficha *p)
{
 int i;

for(i=0;i<=1;i++)
  {
    printf(“%s”,p->nombre);
    printf(“%s”,p->apellido);
    printf(“%d”,p->edad);
    printf(“%f”,p->salario);
    p++;
  }
return;
}
```

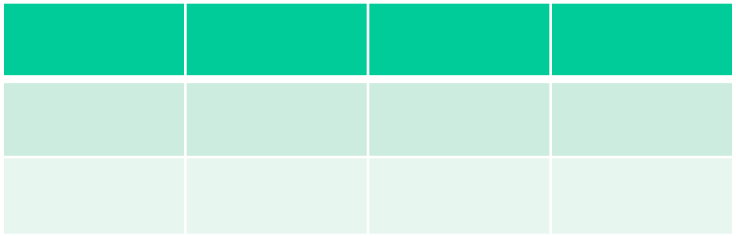
Ejemplo: Pasar una matriz

```
#include<stdio.h>


void imprimir(int matriz[][10]);

int main(){
 int matriz[10][10];
 imprimir(matriz);
 return 0;
}

void imprimir(int matriz[][10]){
 int i, j;
 for (i=0; i<10; i++){
   for (j=0; j<10; j++){
      matriz[i][j]=50;
      printf("%d\t",matriz[i][j]);
  }
  printf("\n");
}
return;
}
```



Otro ejemplo

```
#include<stdio.h>

func(int mat[][3],int f, int c)
/*  func(int (*mat)[3],int f, int c)   es válido recibir así la matriz*/
       /*func(int *mat,int f, int c)   no es válido*/
/*func(int mat[][],int f, int c) no válido*/

{
 int i,j;
 printf("Esta es la matriz");
 for(i=0;i<f; i++)
       {
        for(j=0;j<c; j++)
                printf("%d  ", mat[i][j]);

        printf("\n");
        printf("                  ");
       }
return;
}
int main()
{
 int mat[][3]={1,2,3,4,5,6};
 func(mat,2,3);
return 0;
}
```

manda el puntero a un arreglo

```
#include<stdio.h>
#include<conio.h>

int main()                    /* función que suma un vector */
{
 int a[5]= {10, 20, 30, 40, 50},su;
 int func(int *p);
 su=0;
 su=func(a);
 printf("la suma del vector es =%d", su);
 return;
 }

 int func (int *p)
 {
  int i,sum;
  sum=0;
  for(i=0; i<5; ++i)
       {
         sum+=*(p+i);
       }
  return(sum);
 }
```

Devuelve apuntador a cadena

```
struct datos *copiar (char s[]);
void hacecopia (char v[], struct datos w[]);

int main()
{
      char s[TAM]="Hice una copia";
      char *v;
      v=copiar(s);
      printf("%s \n", v);
      free (v);
      system("PAUSE");
      return 0;
}
```

```
char *copiar (char s[TAM])
{
      char *q;
      q=(char*)malloc(TAM*sizeof(char));
      hacecopia(q,s);

      return q;
}
```

```
void hacecopia (char v[TAM], char w[TAM])
{
      int i;
      for(i=0;i<TAM;i++)
       v[i]=w[i];
}
```